**Detection and tracking of feature points**

**Abstract**

In this report we answer the question of how to choose the feature windows that are best suited for tracking.We also address the issue of how to detect occlusions.

**Introduction**

Two questions must be answered :

1.How to select features.

2.How to track them from frame to frame

The second problem can be solved by a previous result by Lucas and Kanade who proposed a method for registering two images for stereo matching .Their approach is to minimize the sum of squared intensity differences between a past and a current window.

We derive a solution to the 1st problem in this report,we define a feature to be good if it can be tracked well.We also rederive the equations of Lucas and Kanade in a physically intuitive way.

**Feature Tracking**

As the camera moves the patterns of image intensities change .In general,any function of three variables I(x,y,t) ,where x and y are space variables and t is time variable,represents an image sequence.Images taken at near instants satisfies the following property.

I(x, y, t + τ ) = I(x − ξ, y − η, t)

A later image taken at time t + τ can be obtained by moving every point in the current image taken at time t. The amount of motion d = (ξ, η) is called the displacement of the point at x = (x, y) between time instants t and t + τ , and is in general a function of x, y, t, and τ .

The above property is violated at occluding boundaries.In this points do not move within the image but appear and disappear.

**The Approach**

In finding the displacement d of a point from one frame to the next a single pixel cannot be tracked unless it has a very distinctive brightness and also the pixel value can change due to noise and can be confused with adjacent pixels.Because of these problems we consider windows of pixels .The points in the window move at different velocities and may even disappear or appear new.

This is a problem in 2 ways:

1.How do we know that we are following the same window if the contents change over time?

2.How are different velocities combined to give one resulting vector?

Our solution to the 1st problem is residue monitoring:we keep checking that the appearance of a window has not changed too much.If it has,we discard the window.

The second problem could be solved as follows: rather than describing window changes as simple translations, we can model the changes as a more complex transformation, such as an affine map.We choose two parameters for small windows .Any discrepancy between successive windows that cannot be explained by a translation is considered to be error, and the displacement vector is chosen so as to minimize this residue error.

J(x) = I(x, y, t + τ ),

I(x − d) = I(x − ξ, y − η, t),

J(x) = I(x − d) + n(x) ,where n is noise.

The displacement vector d is then chosen so as to minimize the residue error defined by the following

€=∫w[I(x-d)-J(x)]2wdx.

Here w is a weighting function and in simplest case it could be set to 1. It can also be Gaussian like function and also can depend on image intensity pattern.

**Solving for Image Displacement:**

When the displacement vector is small, the intensity function can be approximated by its Taylor series

I(x − d) = I(x) − g d ,

Substituting above equation in €

€= ∫w [I(x) − g d − J(x)]2w dx = ∫w (h − g d)2 w dx

where h = I(x) − J(x).

Differentiating the above equation with respect to d and setting it equal to zero gives the minimum error

∫w (h − g d) gw dA=0

(g.d)g=(ggT)d and d is constant within window

(∫w ggT w dA)d = ∫w hgwdA

This is a system of two scalar equations in two unknowns. It can be rewritten as Gd = e , where the coefficient matrix is the symmetric, 2 × 2 matrix G = ∫w ggT w dA , and the right-hand side is the two-dimensional vector e = ∫w (I − J)gw dA .

Gd = e is the basic step of tracking procedure

G can be computed from one frame, by estimating gradients and computing their second order moments.

e, on the other hand can be computed from the difference between the two frames, along with the gradient computed above.

**Physical Interpretation**

Consider the intensity function within the window W.Make a second copy of it and superimpose on it .There is no space between the two intensity surfaces.If you now move the copy by a small horizontal displacement ,a gap forms between the two surfaces.

The width of the gap, measured horizontally, is a function of the displacement between the two intensity patches. When measured vertically, on the other hand, the width of the gap is just the difference between the values of the two intensity profiles.

We show that for small displacements the horizontal and the vertical width of the gap at a given point in the image are related to each other through the image gradient at that point.

We then look for the displacement that makes the difference between the two expressions as small as possible in the Least Squared Error sense and over the entire window W.This yields the expression for the residue €.

The displacement vector d is in general in a different direction than the image gradient g = ( ∂I ∂x , ∂I ∂y ). If the gradient is expressed as g = gu , where g is the magnitude of g and u is a unit vector, then the displacement ∆ measured along the gradient direction is the projection of d along u: ∆ = d · u .

We see that the vertical gap width h = I − J is h = ∆ tan α ,

where α is the maximum slope of the patch. Since the tangent of α is equal to the magnitude g of the gradient, we can write

h = ∆ g = d · u g = d · g .

By equating the first and last term, we obtain the following equation relating the image gradient g, the inter-frame displacement d, and the difference h between image intensities:

g · d = h

The image gradient g can be estimated from one image,while the difference h is easily computed from both.

The best value for d can be chosen as the one that minimizes the square of that difference, integrated over the entire window. In other words, we minimize the weighted residue

∫w (h − g d)2 gw dA w.r.t d.

**Feature Selection**

Researchers have proposed to track corners or windows with a high spatial frequency content or regions where 2nd order derivatives was high.All these definitions usually yield trackable features but come with no guarantee of being the best for tracking algorithm.So we base our definition on the method we use for tracking.A good window is one that can be tracked well.We can track a window from frame to frame if the system represents good measurements .

This means that 2x2 coefficient matrix G must be both above the image noise level and well conditioned.

Noise requirement implies that both eigen values of G must be large and conditioning requirement means that they cannot differ by several orders of magnitude.

Two small eigenvalues mean a roughly constant intensity profile within a window. A large and a small eigenvalue correspond to a unidirectional pattern. Two large eigenvalues can represent corners, salt-and-pepper textures, or any other pattern that can be tracked reliably.

In practice, when the smaller eigenvalue is sufficiently large to meet the noise criterion, the matrix G is usually also well conditioned. This is due to the fact that the intensity variations in a window are bounded by the maximum allowable pixel value, so that the greater eigenvalue cannot be arbitrarily large.

As a consequence, if the two eigenvalues of G are λ1 and λ2, we accept a window if

min(λ1, λ2) > λ

where λ is a predefined threshold.

To determine λ, we first measure the eigenvalues for images of a region of approximately uniform brightness, taken with the camera to be used during tracking. This gives us a lower bound for λ. We then select a set of various types of features, such as corners and highly textured regions, to obtain an upper bound for λ.

**Experiment**

This experiment deals with the performance of both feature selection and tracking on real images.We use a stream of 100 frames showing surfaces of several different types :a furry puppet,a cylindrical and glossy mug with strong surface markings,an artichoke ,a flat model street sign .Between frames ,the camera was translated to right producing a displacement of about one pixel per frame.

Feature Selection

We take a window of size 15x15 in the first frame and we select a threshold value of 10.We take smaller of the two eigen values(minor eigenvalue) of the tracking matrix G for all square windows .The feature selection algorithm sorts the minor eigenvalues in decreasing order and picks feature coordinates from the top of the sorted list.Every time a coordinate pair is selected ,it is assigned a new feature number.To obtain non overlapping features ,all the features in the list that overlap the window centered at the selected pair are deleted.The requirement of zero overlap can be relaxed by enforcing a minimum distance between window centers smaller than the window size.

We see that the eigen value criterion selects the corners on the mug as well as fuzzier features on the puppet.Also a number of features is found on the artichoke where the intensity patterns are very irregular.No features are found on the background ,uniform areas of the pedestrian sign and on the mug ,and also along the straight edges of the mug.

Tracking

As seen from figure the last of the 100 frames in the sequence,with the superimposed features, as tracked by the algorithm. Each feature required typically fewer than five iterations of the basic tracking step. 217 of the 226 features selected in the first frame survive tracking throughout the stream. No gross errors are made for any of the surviving features. Of the nine missing features at the end of the stream, six disappear off the right image boundary. Of the other three, two (201 and 207, on the fur of the puppet) are too weak to be tracked. The ninth missing feature, number 79, on the right side of the mug is lost because in frame 40 the tracker did not converge within ten iterations. It would have taken 14 iterations for complete convergence, that is, to bring the change in displacement due to a new iteration below one hundredth of a pixel. The reason for the large number of iterations is that feature 79 is on top of a glossy surface viewed at a substantial slant angle. This causes the feature window to change its appearance substantially from frame to frame. During tracking, a cumulative residue is computed for each feature window. This residue is defined as the root mean-squared intensity difference between the first and the current window. The cumulative residue is plotted as a function of the frame number. Notice that most of the residue curves grow at the rate of about one intensity level per pixel every one-hundred frames. A larger residue may indicate occlusion.

Window Size and Occlusion

Smaller windows are more sensitive to noise, however they are affected by distortions due to changes of view point. To illustrate this point, we compared tracking of feature number 2 with square windows 15 and 31 pixels wide. Feature number 2 is the head of the pedestrian on the sign.

A large window is more likely to change during camera motion. Here the boundary of artichoke appears within the large 31x31window of feature 2 somewhere between frame 1 and 100 causing the error. So it is adequate to select the window size which is not very small and also not too large.

Small windows minimize occlusion problems. On the other hand, they will occur no matter what the size of the window. The dashed line in figure 5.9 suggests a threshold on the cumulative residue for the detection of occlusions. Of the features above the threshold, those numbered 7, 8, 12, 16, 17, and 97 are occlusions. The other features, numbered 1, 4, 30, are in an area of the mug that receives strong reflections from the light source. As a result, the overall intensity pattern changes substantially from the first to the last frame, increasing the value of the residue even if the features are tracked well. This simple occlusion detection method would identify most occlusions

False features:

Other occlusion phenomena produce problems that are more difficult to detect. Feature number 45 starts at the intersection of the right boundary of the artichoke with the upper left edge of the traffic sign. As the camera moves, the local appearance of that intersection does not change, but its position in space slides along both edges. The tracker cannot notice the problem, but the feature would create a bad measurement for any motion and shape method that assumes that features correspond to static points in the environment. However, this problem can be detected in three dimensions, after the motion and shape algorithm has been applied